On the Complexity of Multi-Round Divisible Load Scheduling

Yang Yang, Henri Casanova, Maciej Drozdowski, Marcin Lawenda, Arnaud Legrand

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On the Complexity of Multi-Round Divisible Load Scheduling

Yang Yang — Henri Casanova — Maciej Drozdowski — Marcin Lawenda — Arnaud Legrand

N° 6096

Janvier 2007

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6096--FR+ENG

RHÔNE-ALPES

On the Complexity of Multi-Round Divisible Load Scheduling

Yang Yang [*] , Henri Casanova [†] , Maciej Drozdowski [‡] , Marcin Lawenda [§] , Arnaud Legrand [¶]

Thème NUM — Systèmes numériques
Projets Mescal

Rapport de recherche n° 6096 — Janvier 2007 — 23 pages

Abstract: In this paper we study master-worker scheduling of divisible loads in heterogeneous distributed systems. Divisible loads are computations that can be arbitrarily divided into independent "chunks", which can then be processed in parallel. In multi-round scheduling load is sent to each worker as several chunks rather than as a single one. Solving the divisible load scheduling (DLS) problem entails determining the subset of workers that should be used, the sequence of communication to these workers, and the sizes of each load chunk. We first state and establish an optimality principle in the general case. Then we establish a new complexity result by showing that a DLS problem, whose complexity has been open for a long time, is in fact NP-hard, even in the one-round case. We also show that this problem is pseudopolynomially solvable under certain special conditions. Finally, we present a deep survey on algorithms and heuristics for solving the multi-round DLS problem.

Key-words: divisible load, multi-round, linear programming, complexity, optimality principle

This paper is based upon work partially supported by the National Science Foundation under grant number 0234233.

[*] Dept. of Computer Science and Engineering, University of California, San Diego, U.S.A.
[†] Information and Computer Sciences Department, University of Hawai'i at Manoa, Honolulu, U.S.A.
[‡] Institute of Computing Science, Poznań University of Technology, Poznań, Poland
[§] Poznań Supercomputing and Networking Center, Poznań, Poland
[¶] CNRS, ID Laboratory, MESCAL INRIA Project Grenoble, France

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 —Télécopie +33 4 76 61 52 52

Sur la complexité de l'ordonnancement en plusieurs tournées de tâches divisibles

Résumé : Dans cet article, nous nous intéressons à l'ordonnancement maître esclave de tâches divisibles. Une tâche divisible est un calcul pouvant être arbitrairement découpé en sous-tâches indépendantes et pouvant donc être traitées en parallèle. Dans un ordonnancement en plusieurs tournées, chaque esclave reçoit ses sous-tâches en plusieurs fois plutôt qu'en une seule, ce qui permet un meilleur recouvrement des communications par des calculs. Un ordonnancement en plusieurs tournées est caractérisé par le sous-ensemble d'esclaves utilisés, l'ordre dans lequel les communications vers ces esclaves sont effectuées, et la taille de chacune des sous-tâches. Nous établissons des résultats de complexité originaux sur ce problème. Nous énonçons et démontrons un principe d'optimalité pour le cas général. Nous montrons la NP-complétude même dans le cas en une seule tournée. Nous proposons également un algorithme pseudo-polynomial pour certaines situations. Nous montrons que dans toute sa généralité, il est difficile de montrer que ce problème est dans NP et nous faisons un état de l'art des différentes techniques (exactes, garanties ou non-garanties) pour résoudre ce problème.

Mots-clés : tâches divisibles, plusieurs tournées, programmation linéaire, complexité, principe d'optimalité

1 Introduction

The problem of assigning the tasks of a parallel application to distributed computing resources in time and space in a view to optimizing some metric of performance is termed the "scheduling problem". It has been studied for a variety of application models. Popular models include the directed acyclic task graph model [24], and the simpler independent task model in which there is no precedence or communication among computational tasks [13]. These models are representative of many applications in science and engineering. Typically the number of tasks, their communication and computation costs, are set in advance. The scheduling problem is known to be difficult [22]. For instance, in the independent task model, the scheduling problem is akin to bin-packing and, as a result, many heuristics have been proposed in the literature (see [15] for a survey). Another flavor of the independent tasks model is the one in which the number of tasks and the task sizes can be chosen arbitrarily. In this case, the application consists of an amount of computation, or *load*, that can be arbitrarily divided into any number of independent pieces, or *chunks*. In practice, this model is an approximation of an application that consists of a large number of identical, low-granularity units of load. This *divisible load* (DL) model arises in practice in many domains [18, 11, 25, 27, 14, 6, 21, 30] and has been widely studied in the last decade [9, 10, 29]. This paper focuses on the Divisible Load Scheduling (DLS) problem.

We consider distributed computing platforms in which the compute nodes are interconnected in a logical *star topology* (i.e., a single-level tree), which is a popular and realistic model for deploying DL applications in practice. The application runs in master-worker fashion, i.e., the root of the star (the master) initially holds all application input data and dispatches work to the leaves of the star (the workers). We make the common assumptions [9] that the master sends data to only one worker at a time (i.e., the "one-port" model), and that workers can compute and communicate simultaneously (i.e., the "with front-end" model). We focus on heterogeneous platforms, meaning that the communication and computation rates of different workers can be different. Computing a DL schedule entails three steps: (i) select which workers should participate in the computation; (ii) decide in which order workers should receive load chunks and how many times; and (iii) compute how much work each load chunk should comprise. Previously proposed solutions to the DLS problem fall into two categories: *one-round* schedules and *multi-round* schedules. In one-round schedules, each worker receives only one load chunk. In multi-round schedules, each worker may receive multiple load chunks throughout application execution.

Multi-round schedules have been shown to be preferable to one-round schedules because they allow for better overlap of computation and communication [2]. Unfortunately, designing multi-round DLS algorithms is more challenging and fewer results are available in the literature. One key difficulty with multi-round scheduling is due to the presence of *start-up costs*, that is fixed amounts of time that must be spent when sending data over a network. It is known that while one could model the time to send some data over a network as linear in terms of the data size. A better model is to view communication delay as affine in the data size, with a constant portion that corresponds to the overhead of initiating a network connection and to the physical network latency. Modeling this start-up cost is important to be relevant to practice, especially as computing platforms that span wide-area networks have emerged and are prime candidates for loosely-coupled applications such as DL applications [20]. Furthermore, modeling data transfer costs as linear in data size leads to schedules that divides the load into an infinite number of chunks that each need to be sent to workers. Such a schedule would lead to infinite overhead in practice. With start-up costs, the scheduling algorithm must pick an optimal finite number of chunks, which is difficult. Furthermore, without start-up costs, all workers can be utilized as there is no penalty for using even a slow worker. With start-up costs however, the scheduling algorithm must pick workers carefully. Thus, while modeling start-up costs is more relevant to practice, it makes DLS significantly more challenging.

In this paper we focus on the multi-round DLS problem on heterogeneous star networks with communication start-up costs and we make the following contributions:

1. We give the first proof, to the best of our knowledge, of the NP-hardness of the multi-round DLS problem. This result is obtained by first proving NP-completeness for the one-round case, which is a novel result as well.

2. We propose a pseudo-polynomial algorithm to solve the multi-round DLS problem in a particular case.

3. We give an algorithm that computes the optimal solution to the multi-round DLS problem (in exponential time). This algorithm relies on a Mixed Integer Linear Programming (MILP) formulation.

4. We conduct an experimental evaluation, using simulation, of previously proposed heuristics. In particular, for each heuristic we quantify the trade-off between the time to compute the schedule and the quality of the schedule. To the best of our knowledge, this is the most complete such evaluation to date, both in terms of the heuristics and of the range of experimental scenarios.

This paper is organized as follows. We give the NP-completeness proof and the pseudo-polynomial algorithm in Section 4. Section 5 describes the MILP algorithm, while Section 6 highlights previously proposed heuristics. Section 7 summarizes our results and discusses future directions.

2 Problem Definition and Notations

Consider a DL application that consists of W independent units of load to be processed. The processing of each load unit involves performing some computations on some input data. Initially, input data are located on a *master* computer. Without loss of generality, we assume that the master does not perform any computation. The master can send input data for one or more load units to p *workers*. Worker i can process a chunk of x load units in xA_i seconds, and the master can send a chunk of x load units to worker i in $S_i + xC_i$ seconds. We assume that the A_i's, S_i's, and C_i's are integer, while x and W are rational. We assume that the master cannot send chunks to more than one worker at a time, following the one-port model. We also assume that a worker may compute and receive data simultaneously. However a worker has to wait for a chunk to be completely transfered before starting processing it. Both these assumptions are commonly used in the DLS literature. We do not consider transfer of output data back to the master. This is also a common assumption in the literature (interested readers can find a discussion of output data in [9, 18, 31]).

The problem we consider in this paper is: how should the master partition the load into chunks and send those chunks to the workers so that the application makespan, i.e., the time at which the last unit of load is completed, is minimized? A schedule consists of a sequence of workers to which the master sends load chunks in order, which is called the *activation sequence*, and the size of each load chunk. In the rest of the paper we denote by $\alpha_i^{(j)}$ the size of the j^{th} chunk of load sent to worker i, measured as a rational number of load units. α_i denotes the size of the chunk of load sent to worker i in case only one chunk is sent to worker i in the schedule. Some workers may not be used in the schedule and do not appear in the activation sequence. In the following, we will denote as act_{max}, the largest number of activations allowed in an activation sequence. In the one-round case, a worker can only appear once in the activation sequence. The typical notion of multi-round used in the literature assumes that the activation sequence is periodic. Hence, if we denote by r_{max} the largest number of rounds allowed in a multi-round schedule and by r_{size} the number of processors used in each round, we have $act_{max} = r_{max} r_{size}$. In this paper we impose no periodicity constraints on the activation sequence and instead consider the general case (e.g., some workers may appear twice as often as other workers in the activation sequence).

We define the associated decision problem as:

Problem 1 (DLS). *Given p workers, $(A_i)_{1 \leqslant i \leqslant p}$, $(S_i)_{1 \leqslant i \leqslant p}$, $(C_i)_{1 \leqslant i \leqslant p}$, and two rational numbers $W \geqslant 0$ and $T \geqslant 0$, is it possible to compute all W load units within T seconds after the master starts sending out the first load unit?*

In the following, we may consider some restrictions of the DLS problem. These restrictions will be denoted as DLS{restriction} where restriction may be for example: *1Round* (all processors are used at most one time), $C_i = 0$ (bandwidths from the master to the slaves are infinite), $S_i = 0$ (no latency), and so on.

Similarly, we define the two optimization problems:

Problem 2 (DLS-Opt$T$). *Given a rationnal workload W, p workers, $(A_i)_{1 \leqslant i \leqslant p}$, $(S_i)_{1 \leqslant i \leqslant p}$, $(C_i)_{1 \leqslant i \leqslant p}$, what is the smallest rational number $T \geqslant 0$ such that is it possible to compute all W load units within T seconds after the master starts sending out the first load unit?*

Problem 3 (DLS-Opt$W$). *Given a rationnal time bound T, p workers, $(A_i)_{1 \leqslant i \leqslant p}$, $(S_i)_{1 \leqslant i \leqslant p}$, $(C_i)_{1 \leqslant i \leqslant p}$, what is the largest rational number $W \geqslant 0$ such that is it possible to compute all W load units within T seconds after the master starts sending out the first load unit?*

3 Optimality Principle

In this section, we discuss the *Optimality Principle* first proposed by Robertazzi [9]. We start by recalling how DLS{FixedActivation}, DLS{FixedActivation}-Opt$T$, and DLS{FixedActivation}-Opt$W$ can be solved in polynomial time and then we discuss the precise formulation of the Optimality Principle. Last, we prove this optimality principle.

3.1 DLS{FixedActivation} and Linear Programming

Consider a given instance $I = (S, C, A)$ of the problem. Let $\sigma : \{1, \ldots, n\} \to \{1, \ldots, p\}$ denote a given activation sequence of size n. Then if we denote by α_j the amount of workload sent to $P_{\sigma(j)}$, DLS{FixedActivation} is equivalent to determining whether the following linear constraints define a non-empty set:

$$\begin{cases} \text{(1a)} & \displaystyle\sum_{j=1}^{n} \alpha_j = W \\ \text{(1b)} & \forall k \leqslant n : \displaystyle\sum_{j=1}^{k} \left(S_{\sigma(j)} + \alpha_j C_{\sigma(j)}\right) + \sum_{j \geqslant k \,:\, \sigma(j) = \sigma(k)} \alpha_j C_{\sigma(k)} \leqslant T \\ \text{(1c)} & \forall j \leqslant n : \alpha_j \geqslant 0 \end{cases} \tag{1}$$

The leftmost part of Constraint (1b) represents the time at which the k^{th} communication ends and the middle one is a lower bound on the computation time of worker $\sigma(k)$ after this communication. The sum of these two times has thus to be smaller than the makespan T. Considering in backward order the activations where a given worker l is used, it is not hard to see from the constraints that one will obtain a feasible schedule [16].

Likewise, DLS{FixedActivation}-Opt$T$ is equivalent to the following linear program:

$$\begin{array}{l} \textsc{Minimize} \quad T \quad , \\ \textsc{under the constraints} \\ \begin{cases} \text{(2a)} & \displaystyle\sum_{j=1}^{n} \alpha_j = W \\ \text{(2b)} & \forall k \leqslant n : \displaystyle\sum_{j=1}^{k} \left(S_{\sigma(j)} + \alpha_j C_{\sigma(j)}\right) + \sum_{j \geqslant k \,:\, \sigma(j) = \sigma(k)} \alpha_j C_{\sigma(k)} \leqslant T \\ \text{(2c)} & \forall j \leqslant n : \alpha_j \geqslant 0 \end{cases} \end{array} \tag{2}$$

and DLS{FixedActivation}-Opt$W$ is equivalent to the following linear program:

$$\begin{array}{l} \textsc{Maximize} \quad W = \sum_{j=1}^{n} \alpha_j \quad , \\ \textsc{under the constraints} \\ \begin{cases} \text{(3a)} & \forall k \leqslant n : \displaystyle\sum_{j=1}^{k} \left(S_{\sigma(j)} + \alpha_j C_{\sigma(j)}\right) + \sum_{j \geqslant k \,:\, \sigma(j) = \sigma(k)} \alpha_j C_{\sigma(k)} \leqslant T \\ \text{(3b)} & \forall j \leqslant n : \alpha_j \geqslant 0 \end{cases} \end{array} \tag{3}$$

We can define the two functions:

$$W_\sigma : \begin{cases} [\sum_{i=1}^{n} S_{\sigma(i)}, \infty[\to [0, \infty[\\ \qquad T \mapsto \sup\{W | \mathsf{DLS}(I, \sigma, W, T) \text{ has a solution}\} \end{cases}$$

$$T_\sigma : \begin{cases} [0, \infty[\to [\sum_{i=1}^{n} S_{\sigma(i)}, \infty[\\ \qquad W \mapsto \inf\{T | \mathsf{DLS}(I, \sigma, W, T) \text{ has a solution}\} \end{cases}$$

Theorem 1. *W_σ and T_σ are continuous piecewise-linear functions and are inverse of each other.*

Proof. Liner program (3) can be written:

$$\begin{array}{l} \textsc{Maximize} \quad \sum_{j=1}^{n} \alpha_j \quad , \\ \textsc{under the constraints} \\ \begin{cases} \forall k \leqslant n : (B_\sigma \alpha)_k \leqslant T - c_k \\ \forall j \leqslant n : \alpha_j \geqslant 0 \end{cases} \end{array} \tag{4}$$

which has the same optimal value as its dual linear program

$$\begin{array}{l} \text{MINIMIZE} \quad \sum_{k=1}^{n}(T-c_k)y_k \quad , \\ \text{UNDER THE CONSTRAINTS} \\ \begin{cases} \forall k \leqslant n : (B_\sigma^T y)_k \geqslant 1 \\ \forall k \leqslant n : y_k \geqslant 0 \end{cases} \end{array} \tag{5}$$

Let us denote by $\mathcal{P}_\sigma = \{y | \forall k \leqslant n : (B_\sigma^T.y)_k \geqslant 1 \text{ and } y_k \geqslant 0\}$. $\mathcal{P}_\sigma$ is a convex polyhedron and we know that optimal solutions to the linear programs are found on the vertices of their polyhedron. Let us denote by $\delta\mathcal{P}_\sigma$ the set of vertices of $\mathcal{P}_\sigma$. Therefore we have:

$$W_\sigma(T) = \min\left\{\sum_{k=1}^{n}(T-c_k)y_k | y \in \delta\mathcal{P}_\sigma\right\}$$

Note that $\delta\mathcal{P}_\sigma$ is finite and does not depend on T. As for each $y \in \delta\mathcal{P}_\sigma$, $T \mapsto \sum_{k=1}^{n}(T-c_k)y_k$ is an affine function, W_σ is thus the minimum of affine functions, hence a continuous piecewice-linear function. As W_σ is strictly increasing, W_σ is a bijection and T_σ is its inverse function. T_σ is hence also a continuous piecewice-linear function. □

Corrolary 2. *W_{opt} and T_{opt} are continuous piecewise-linear functions and are inverse of each other.*

3.2 Stating the Optimality Principle

Linear programs as (2), (3) will therefore be used in many heuristics that only come up with an activation sequence (e.g., the heuristics of Section 6.1). This approach differs from the solution of [9] where it is assumed that optimal sequence have no idle times, i.e. that selected processors start working as soon as they receive their first chunk and keep working until the end of the schedule. In other words, they compute all the time and all stop computing at the same time. This assumption simplifies linear inequalities (1b) into linear equations. Thus, the formulations (2) and (3) reduce to a system of linear equations, which can be solved in $O(n)$ time due to the particular structure of the system. The fact that the optimal sequence has no idle time is known in the literature under the name of "Optimality Principle" [9]. However, for an arbitrary activation sequence, this assumption does not hold true: there may be idle times (see Figure 1). Figure 1(b) proves that there exists an activation sequence such that the optimal load distribution for DLS-Opt$T$ or DLS-Opt$W$ has idle time. However, one may argue that in this example processor 1 does not work at all and the optimality principle could then be reformulated as:

Proposition 3. *For a given activation sequence, in an optimal load distribution (both for DLS{FixedActivation}-Opt$W$ and for DLS{FixedActivation}-Opt$T$), either a processor has no idle time or it does not receive any load.*

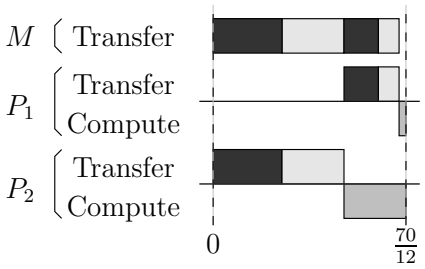
Unfortunately, this formulation does not hold either, as shown by the Figure 1(c). For an arbitrary activation sequence, a processor may receive some load and have idle time. Thus there is no hope to prove that an optimality principle hold for an arbitrary sequence. However, we can check that on this specific instance, the optimality principle holds for the optimal activation sequence (see Figure 1(d)). In the next section, we will prove optimality principle for optimal activation sequences.

Close results have been proved in the past. For example, it has been proved for the one-round problem in [2] that in the optimal selection, all selected workers finish computing at the same time. For multi-round schedules on identical processors it has been shown [28] that there are no idle times neither in the communications nor in the computations.

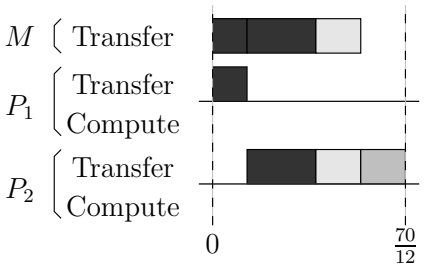
3.3 Proving the Optimality Principle

In this section, we state and prove the optimality principle for optimal activation sequences and DLS-Opt$W$ using similar ideas as [2].

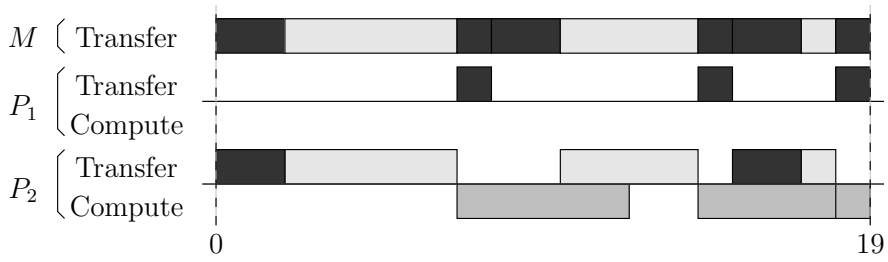
Theorem 4 (Optimality Principle)**.** *For an optimal activation sequence and the corresponding optimal load distribution in DLS-Opt$W$ and DLS-Opt$T$, all messages, except maybe the trailing ones, convey some load and there is no idle time.*



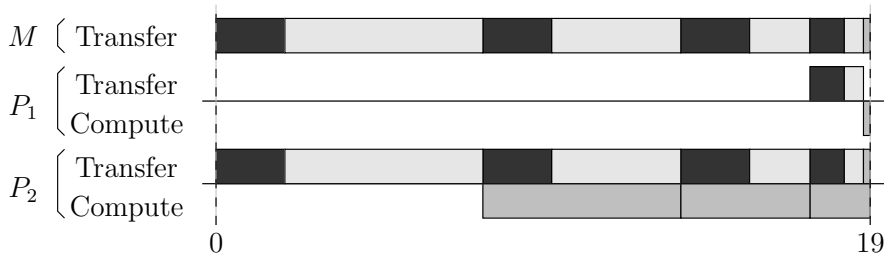
(a) Optimal load distribution for $\sigma = (2, 1)$ and $T = 70/12$: $W_{opt} = 2$ and all processors stop working at the same time.



(b) Optimal load distribution for $\sigma = (1, 2)$ and $T = 70/12$: $W_{opt} = \frac{17}{12}$. Processor 1 does not work at all and thus finishes before time T.

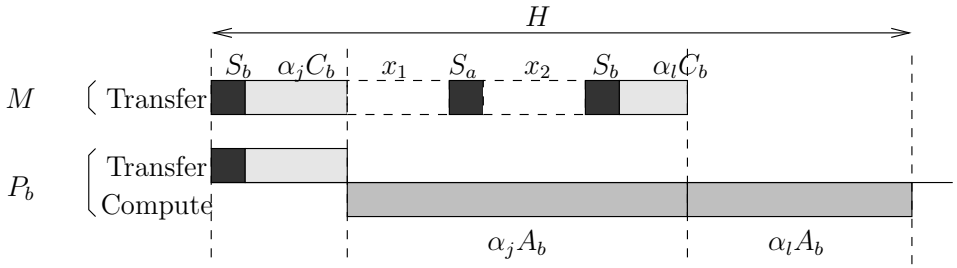


(c) Optimal load distribution for $\sigma = (2, 1, 2, 1, 2)$ and $T = 19$: $W_{opt} = 10$. Processor 2 receives some load and has idle time.

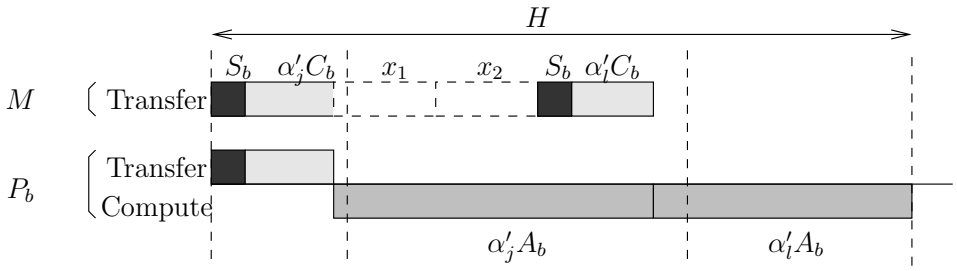


(d) Optimal activation sequence ($\sigma = (2, 1, 2, 1, 2)$) and optimal load distribution for $T = 19$: $W_{opt} = \frac{249}{22}$. All processors receive some load and there is no idle time.

Figure 1: DLS{FixedActivation}-Opt$W$ for $(S_1, C_1, A_1) = (1, 10, 1)$, $(S_2, C_2, A_2) = (2, 1, 1)$.



(a) Initial Situation



(b) Decreasing initial chunks so that the communication gap disappears

Figure 2: Closing the gap after an empty message.

Proof. Let us consider an optimal activation sequence σ for some instance I of DLS-Opt$W$. Note that if there exists processor P_i such that $S_i < T - \sum_{j=1}^{n}(S_{\sigma(j)} + \alpha_j C_{\sigma(j)})$ then it is possible to append a message to P_i with no load to the original communication sequence σ without increasing schedule length T or violating the original schedule. Such an activation sequence is not minimal. In the further discussion we assume that σ has no trailing empty messages. We assume that σ is not empty, and is feasible for the given T, i.e. $\sum_{j=1}^{n} S_{\sigma(j)} < T$.

For a given σ DLS{FixedActivation}-Opt$W$ is equivalent to linear program (3). Let us call α the corresponding load distribution. It is known that the optimal solutions of linear programs are either on vertices of the polyhedron defined by the linear constraints, or a whole facet of the polyhedrons.

Assume that the optimum solution of the linear program (3) is in the corner of the polyhedron. (3) has n variables and $2n$ constraints. Therefore, at least n constraints amongst $2n$ are equalities.

- If none of the constraints (3b) is an equality in the optimal solution then all the constraints (3a) are equalities, which means that there is no idle time, and all messages convey some load.
- If $l > 0$ constraints (3b) are equalities, then $n-l$ constraints (3a) are equations, and l messages carry no load. The $n-l$ remaining non-zero α_j s satisfy, $n-l$ constraints (3a) which are equations. Hence, there are idle times neither in the communications to nor in the computations of the processors receiving any load.

  The l messages with no load contributed only some startup times in the communications. We will show that by removing the startup times of the l messages with no load from the schedule, W can be increased without increasing schedule length T which will contradict the assumption that sequence σ and its distribution are optimum for T.

  Consider an empty message k directed to some processor P_a. Suppose there is some other processor P_b with two nonempty messages which enclose the empty message to P_a (cf. Figure 2(a)). Thus, there are messages $j < k < l$ such that $\sigma(j) = \sigma(l) = b$, and $\alpha_j > 0, \alpha_l > 0$. Let x_1 be the duration of the messages which follow message j, and precede message k. Let x_2 be the duration of the messages which follow k, and precede l. Together $x_1 + x_2 = x$. Let H be the length of the interval since the beginning of communication j till the end of the computation l. Since there were no idle times in the communications and computations on the processors which received non-zero load we can observe that:

$$\begin{aligned} \alpha_j A_b &= x + S_a + S_b + \alpha_l C_b \\ H &= S_b + \alpha_j C_b + x + S_a + S_b + \alpha_l C_b + \alpha_l A_b \end{aligned}$$

From which we obtain:

$$\alpha_j = \frac{HC_b + S_bA_b + (x+S_a)A_b - S_bC_b}{A_b^2 + A_bC_b + C_b^2}$$
$$\alpha_l = \frac{HA_b - 2S_bA_b - (x+S_a)(A_b+C_b) - S_bC_b}{A_b^2 + A_bC_b + C_b^2}$$

and from the above:

$$\alpha_j + \alpha_l = \frac{H(A_b+C_b) - S_bA_b - (x+S_a)C_b - 2S_bC_b}{A_b^2 + A_bC_b + C_b^2}$$

The interval of startup time can be closed by increasing size of message l. Thus, a new schedule can be constructed such that there are no idle times in the communications and computations on P_b (cf. Figure 2(b)). Analogously to the previous reasoning we have:

$$\alpha'_jA_b = x + S_b + \alpha'_lC_b$$
$$H = S_b + \alpha'_jC_b + x + S_b + \alpha'_lC_b + \alpha'_lA_b$$

From which we obtain:

$$\alpha'_j = \frac{HC_b + S_bA_b + xA_b - S_bC_b}{A_b^2 + A_bC_b + C_b^2} \qquad \alpha'_l = \frac{HA_b - 2S_bA_b - x(A_b+C_b) - S_bC_b}{A_b^2 + A_bC_b + C_b^2}$$
$$\alpha'_j + \alpha'_l = \frac{H(A_b+C_b) - S_bA_b - xC_b - 2S_bC_b}{A_b^2 + A_bC_b + C_b^2}.$$

The amount of processed load increased by $\alpha'_j + \alpha'_l - \alpha_j - \alpha_l = \frac{S_aC_b}{A_b^2+A_bC_b+C_b^2} > 0$. In the new schedule communication j with size α'_j finishes earlier by $\frac{A_bC_bS_a}{A_b^2+A_bC_b+C_b^2} > 0$ units of time. Analogously, message l in the new schedule with size α'_l finishes $S_a\left(1 - \frac{C_b^2}{A_b^2+A_bC_b+C_b^2}\right) > 0$ units of time earlier than in the old schedule. Hence, the new schedule does not delay initiation of the computation on any other processor because the new messages j, l are finishing earlier than in the initial schedule. Thus, schedule length does not increase, but the amount of processed load increased which contradicts the assumption that σ is optimum and W is maximum for the given T.

Assume that the empty message k is not enclosed by two nonempty messages to any processor. Since σ is nonempty, message k is either preceded or succeeded by the nonempty message(s). Suppose message is only followed by nonempty messages. By shifting a wole schedule by $S_{\sigma(k)}$ units of time earlier, we still get a valid solution for $\mathsf{DLS}(W, T - S_{\sigma(k)})$. As W_{opt} is strictly increasing, it is possible to perform strictly more work in time T than W, which contradicts the assumption that σ is optimum. Last, suppose there are no nonempty messages after k. This contradicts the assumption that the trailing messages are nonempty.

Consequently, the optimum sequence and the corresponding optimal load distribution have no empty messages, and idle times neither in communications nor in computations.

Thus, we know that the only optimal vertices are the ones such that all the constraints (3a) are equalities. As there is only *one* such vertex, the facet case cannot happen.

We have just proved that the optimality principle holds true for DLS-Opt$W$. As DLS-Opt$W$ and DLS-Opt$T$ are inverse of each other, the optimality principle also holds true for DLS-Opt$T$. □

4 Complexity of Multi-Round DLS

4.1 Previously Obtained Complexity Results

Previous works have studied the complexity of the DLS problem on a heterogeneous star platform with affine communication costs.

A result given by Bharadwaj et al. [7] states that DLS{1-round,S_i =0} can be solved by sorting processors by increasing C_i's in the activation sequence. The assumption that all S_i's are equal to zero does however not hold in practice. This is why Blazewicz and Drozdowski introduce in [12] DLS problems

in which communication incurs a start-up cost (i.e., $S_i \neq 0$), and they solve them in certain special cases. Unfortunately, the general DLS problem is difficult due to the need to determine the optimal activation sequence, which is highly combinatorial. In fact, it is known that solving the DLS problem (as well as DLS-Opt$T$ and DLS-Opt$W$) for a given activation sequence can be done with polynomial complexity (e.g. see [4] and Section 3). In [2] it is shown that the difficult activation sequence computation problem can be bypassed if one assumes that the total load is "sufficiently large". In this case, all start-up costs are much smaller than the makespan, and workers should be sorted by increasing C_i's just like when all S_i's are zero. In spite of these results, it is acknowledged that the complexity of the general DLS problem above is open [2].

In [19], the authors study the DLS problem with added "buffer constraints", i.e., for each worker a bounded number of load units can be stored on that worker. This limitation essentially provides one more condition, which helps when reducing from known NP-complete problems [19]. In [3] this result is strengthened by proving that the DLS problem with buffer constraints is NP-complete in the strong sense. In this paper we prove the NP-completeness in the weak sense of the general DLS problem without additional buffer constraints.

4.2 One-Round DLS is NP-Hard

In this section we study one-round schedules, that is the ones in which each worker appears at most once in the activation sequence. Without loss of generality, we assume that the bandwidth between the master and each worker is infinite, i.e., the time to send x load units to worker i is S_i seconds. We now consider the following associated decision problem:

Problem 4 (DLS{1Round, C_i =0}). *Given W, p workers, $(A_i)_{1\leqslant i\leqslant p}$, $(S_i)_{1\leqslant i\leqslant p}$, and a rational number $T \geqslant 0$, and assuming that bandwidths are infinite, is it possible to compute all W load units within T time units?*

We prove that DLS{1Round, C_i =0} is NP-complete. Since DLS{1Round, C_i =0} is a special case of DLS, we obtain the NP-hardness of the more general DLS. DLS{1Round, C_i =0} is difficult because the total communication start-up times, $\sum_{1\leqslant i\leqslant N} S_i$, may be larger than T. Therefore, one must use a carefully chosen *subset* of the workers, which gives the problem a combinatorial flavor. Intuitively, for an instance to satisfy DLS{1Round, C_i =0}, it has to meet two requirements: (i) have a makespan lower than T, meaning that the sum of communication start-up costs of the selected workers must be *small enough*; and (ii) compute more than W_0 units of load, meaning that the compute speeds of the selected workers must be *large enough*. Those two requirements suggest a reduction from the 2-Partition problem. In the reduction from 2-Partition to DLS{1Round, C_i =0}, we have the following variables at our disposal, which can be set freely to "force" the selection of workers: W, T, $(A_i)_{1\leqslant i\leqslant p}$, $(S_i)_{1\leqslant i\leqslant p}$. One must then carefully choose a *small enough* T, and a *large enough* W.

Theorem 5. *DLS{1Round, C_i =0} is NP-Complete.*

Proof. We first show that DLS{1Round, C_i =0} is in NP. A solution to the problem consists of an activation sequence and load chunk sizes. An activation sequence is a string of length at most p. For a given activation sequence, it is known that one can compute the load chunk sizes in polynomial time [4] (see also Section 5.1). Therefore, given a solution to an instance of DLS{1Round, C_i =0}, one can verify in polynomial time that the subset of workers complete workload W within time T.

We prove that DLS{1Round, C_i =0} is NP-complete via a reduction from the NP-complete 2-Partition problem [22], which is defined as follows:

Problem 5 (2-Partition). *Given a finite set B of integers b_i, $1 \leqslant i \leqslant 2m$, is there a subset $B' \subset B$ such that $|B'| = m$ and*

$$\sum_{b_i \in B - B'} b_i = \sum_{b_i \in B'} b_i = L?$$

Given an instance of 2-Partition, we construct an instance of DLS{1Round, C_i =0} as follows. For each b_i we create a worker P_i, with start-up cost S_i and computation speed $\frac{1}{A_i}$ such that $S_i = \frac{1}{A_i} = M + b_i$, for a total of $p = 2m$ workers. We then choose $T = mM + L + \frac{1}{2}$, and $W = \frac{m}{2}(m-1)M^2 + (m-1)LM + \frac{m}{2}M$. We must chose M above as a "large" number; it turns out that choosing $M = 8m^2L^2$ is sufficient.

Figure 3 depicts a schedule with four workers with time on the horizontal axis (from time 0 to time T) and the four workers on the vertical axis. For each worker we show a communication phase (in white),

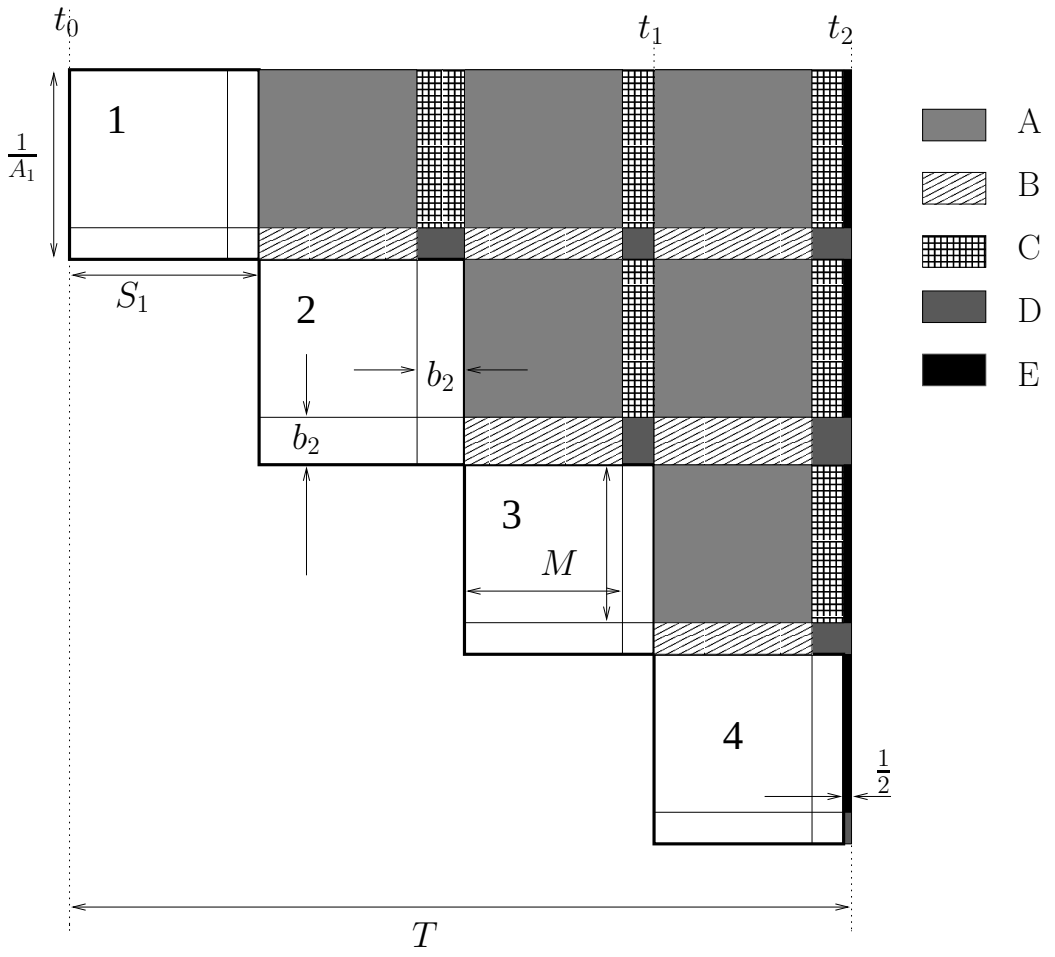


Figure 3: Illustration to the proof of Theorem 5.

followed by a computation phase (in various shades of gray, whose meaning will be explained shortly). Note that all workers finish computing at the same time. Indeed, since we have a fixed cost for sending chunks over the network, one can easily send enough load to each worker to keep it busy until time T. Clearly, the number of load units that a worker contributes to the computation is the product of its computational speed, $\frac{1}{A_i}$, and the duration of the time interval from the end of communication to the overall application finish time. To make the proof simpler to follow, we let the width of each worker slot in the figure be S_i, so that the worker's contribution to the overall computation number of work units is given by the **area** of its slot. In the proof, we will compute such areas in order to estimate numbers of computed load units. We prove the reduction with the usual two steps.

Lemma 6. *2-Partition* $\Rightarrow$ *DLS{1Round, C_i =0}*

Proof. If we have a solution to the 2-Partition problem, we show that we also have a solution to the DLS{1Round, C_i =0} problem. Pick all m workers P_i such that $b_i \in B'$. For simpler notations, and without loss of generality, we assume that $i = 1, \ldots, m$. First, we note that the sum of the communication start-up costs is

$$\sum_{i=1}^{m} S_i = \sum_{i=1}^{m} (M + b_i) = mM + L < T.$$

Consequently, all m workers can participate in the computation and appear in the schedule.

Given the set of workers participating in the schedule, we now estimate the number of load units that are computed before time T, which corresponds to the shaded area shown in Fig. 3. The figure shows the shaded area partitioned in five different types of rectangular zones. Zone A consists of $m(m-1)/2$ squares of dimension $M \times M$. Zone B consists of $m - i$ rectangles of dimension $M \times b_i$ for $i = 1, \ldots, m-1$, for a total of $m(m-1)/2$ such rectangles. Zone C is similar to zone B, but with $i - 1$ rectangles of dimension

$b_i \times M$ for $i = 2 \ldots, m$. Zone D, which visually corresponds to the intersections between rectangles in zones B and C, consists of $m(m-1)/2$ rectangles of dimensions $b_j \times b_i$, for $i = 1, \ldots, m-1$, and $j \in [2, \ldots, m]$ with $j > i$, and of m rectangles of dimension $\frac{1}{2} \times b_i$, for $i = 1, \ldots, m$. Finally, zone E consists of m rectangles of dimension $\frac{1}{2} \times M$.

We compute the sum of the areas of zones A, B, C, and E. The total area of zone A is clearly $M^2 m(m-1)/2$. The total area of zone B is:

$$M((m-1)b_1 + (m-2)b_2 + \ldots + b_{m-1}).$$

The total area of zone C is:

$$M(b_2 + 2b_3 + 3b_4 + \ldots + (m-1)b_m).$$

Finally the total area of zone E is $\frac{m}{2}M$. Because we have not counted the area due to rectangles in zone D, we can bound below the total number of computed load units, $\mathcal{W}$, as follows:

$$\begin{aligned}
\mathcal{W} &\geqslant \frac{m}{2}(m-1)M^2 + M\sum_{i=1}^{m-1} i b_{i+1} + M\sum_{i=1}^{m-1}(m-i)b_i + \frac{m}{2}M \\
&= \frac{m}{2}(m-1)M^2 + (m-1)M\sum_{i=1}^{m} b_i + \frac{m}{2}M \\
&= \frac{m}{2}(m-1)M^2 + (m-1)ML + \frac{m}{2}M \\
&= W.
\end{aligned}$$

We conclude that we have a solution to the DLS{1Round, C_i =0} problem. □

Lemma 7. *DLS{1Round, C_i =0} ⇒ 2-Partition*

Proof. If we have a solution to the DLS{1Round, C_i =0} problem, we now show that we also have a solution to the 2-Partition problem. First, we know that in the solution to DLS{1Round, C_i =0} we cannot have more than m workers. Otherwise, the S_i startup costs would add up to a time larger than T. We can also see that we need at least $m-1$ workers. Indeed, a smaller number of workers will not suffice because the overall number of computed load units will be strictly lower than W. (Intuitively, we waste the opportunity to compute at least M^2 load units when using $m-2$ workers.) Therefore we must use either $m-1$ or m workers. However, we now prove that we cannot use $m-1$ workers.

Let us assume that the solution of DLS{1Round, C_i =0} uses $m-1$ workers, and let us compute the total number of load units computed before time T. The intuition is that by not having the m^{th} worker we miss its contribution in zone E, which makes the overall number of computed load units strictly lower than W. We count the area in two parts, the area before worker $m-1$ finishes communication (left of time instant t_1 in Fig. 3), and the area after that. For the first part, the squares in zone A sum up to $M^2(m-1)(m-2)/2$. Those in zones B and C sum up to $M(m-2)\sum_{i=1}^{m-1} b_i$ as in the previous section of the proof. Rectangles in zone D left of t_1 take up $\sum_{1 \leqslant i < j \leqslant m-1} b_i b_j < \frac{1}{2}(m-1)(m-2)(2L)^2 < 2m^2L^2$. The second part is easily computed as the area between t_1 and t_2:

$$\left(\sum_{i=1}^{m-1} \frac{1}{A_i}\right)\left(T - \sum_{i=1}^{m-1} S_i\right).$$

We can add the two parts and obtain the total number of computed load units, $\mathcal{W}$, as follows:

$$\begin{aligned}
\mathcal{W} &< \left\{\frac{(m-1)(m-2)}{2}M^2 + M(m-2)\sum_{i=1}^{m-1} b_i + 2m^2L^2\right\} + \\
&\quad \left\{\left[(m-1)M + \sum_{i=1}^{m-1} b_i\right]\left[T - \left((m-1)M + \sum_{i=1}^{m-1} b_i\right)\right]\right\} \\
&= \frac{(m-1)m}{2}M^2 + L(m-1)M + 2m^2L^2 + \\
&\quad (L+\frac{1}{2})\sum_{i=1}^{m-1} b_i - (\sum_{i=1}^{m-1} b_i)^2 + \frac{1}{2}mM - \frac{1}{2}M \\
&< \left[\frac{(m-1)m}{2}M^2 + L(m-1)M + \frac{1}{2}mM\right] + \left[\left(2m^2L^2 + (L+\frac{1}{2})\sum_{i=1}^{m-1} b_i\right) - \frac{1}{2}M\right] \\
&< \frac{(m-1)m}{2}M^2 + L(m-1)M + \frac{1}{2}mM \\
&= W.
\end{aligned}$$

Therefore, we cannot have a solution with $m-1$ workers, and we must use exactly m workers.

For the solution of DLS{1Round, C_i =0} with m workers we can write that the makespan is lower than T (otherwise it would not be a solution). The makespan is equal to the sum of the S_i series and T is equal to $mM + L + \frac{1}{2}$. Therefore, we have:

$$\sum_{i=1}^{m} S_i \leqslant mM + L + \frac{1}{2}$$

Replacing S_i by its value, we obtain:

$$\sum_{i=1}^{m} b_i \leqslant L + \frac{1}{2}.$$

Because $\sum_{i=1}^{m} b_i$ is integer, we have

$$\sum_{i=1}^{m} b_i \leqslant L. \tag{6}$$

We now estimate the total number of load units computed. The small area of zone D, denoted by $\mathcal{D}$, is

$$\sum_{1\leqslant i<j\leqslant m} b_i b_j + \frac{1}{2}\sum_{1\leqslant i\leqslant m} b_i \leqslant \frac{1}{2}m(m-1)L^2 + \frac{1}{2}L,$$

because of Eq. 6. Then the total computed load $\mathcal{W}$ is:

$$\begin{aligned}
\mathcal{W} &= \frac{m}{2}(m-1)M^2 + M(m-1)\sum_{i=1}^{m} b_i + \mathcal{D} + \frac{1}{2}mM \\
&\leqslant \frac{m}{2}(m-1)M^2 + M(m-1)\sum_{i=1}^{m} b_i + \frac{1}{2}mM + \frac{1}{2}m(m-1)L^2 + \frac{1}{2}L.
\end{aligned}$$

Since the schedule is a solution of DLS{1Round, C_i =0}, we also have:

$$\mathcal{W} \geqslant W = \frac{m}{2}(m-1)M^2 + (m-1)LM + \frac{1}{2}mM.$$

Therefore

$$M(m-1)\sum_{i=1}^{m} b_i + \frac{1}{2}m(m-1)L^2 + \frac{1}{2}L \geqslant (m-1)LM,$$

which, given that $M = 8m^2L^2$, implies that:

$$\sum_{i=1}^{m} b_i \geqslant L - \frac{1}{16m} - \frac{1}{(m-1)8m^2L} > L - 1.$$

Because $\sum_{i=1}^{m} b_i$ is integer, we have:

$$\sum_{i=1}^{m} b_i \geqslant L. \tag{7}$$

Inequalities (6) and (7) show that $\sum_{i=1}^{m} b_i = L$. Therefore, there is a solution to the 2-PARTITION problem, which is obtained by picking all the b_i values that correspond to the workers participating in the computation in the solution for the DLS{1Round, C_i =0} problem. □

Consequently, DLS{1Round, C_i =0} is weakly NP-complete, showing that the one-round DLS problem is NP-hard. □

Note that in the proof, we never rely on the assumption that the distribution is done in one round. As a consequence the decision problem (DLS{C_i =0}) associated to the *multi-round* DLS problem with infinite bandwidths is NP-complete too, and the multi-round DLS problem is NP-hard. As a matter of fact, in the infinite bandwidths setting, there is nothing to gain in making two communication to a given processor. Hence the optimal solution uses at most one round and DLS{1Round, C_i =0} =DLS{C_i =0}.

4.3 A Pseudo-Polynomial Algorithm

We analyze two dual optimization problems: DLS{1Round, C_i =0}-Opt$T$: Given W find the shortest schedule of length T^*; DLS{1Round, C_i =0}-Opt$W$: Given a schedule of length T find the maximum load W^* that can be processed in this time limit. We will demonstrate that both problems can be solved in pseudopolynomial time if $(C_i)_{1\leqslant i\leqslant p} = 0$. The algorithm we propose solves problem DLS{1Round, C_i =0}-Opt$W$. In a dual problem the optimum schedule length can be found by use of a binary search over values of T. Recall that $(A_i)_{1\leqslant i\leqslant p}, (S_i)_{1\leqslant i\leqslant p}$ are integers, which allows W, T, W^*, T^* to be rational numbers. We first establish several facts.

Proposition 8. *For a given time limit T, and set $\mathcal{P}' \subseteq \{P_1, \ldots, P_p\}$ of workers taking part in the computation the maximum load is processed if workers are ordered according to nondecreasing values of S_iA_i, for $P_i \in \mathcal{P}'$.*

Proof. The proof is based on an interchange argument. Consider two workers P_i and P_j that are consecutive in the activation sequence. Let the communication to the pair start at time $0 \leqslant x \leqslant T - S_i - S_j$. The communications with P_i, P_j are performed in interval $[x, x + S_i + S_j]$. A change in the sequence of P_i, P_j does not influence the schedule for the other workers. The load processed by the two workers in sequence (P_i, P_j) is:

$$W_1 = \frac{T - x - S_i}{A_i} + \frac{T - x - S_i - S_j}{A_j}.$$

For sequence (P_j, P_i) the load processed by the two workers is

$$W_2 = \frac{T - x - S_j}{A_j} + \frac{T - x - S_j - S_i}{A_i}.$$

From which we get

$$W_1 - W_2 = \frac{S_j}{A_i} - \frac{S_i}{A_j}.$$

Thus, the load is greater for the sequence (P_i, P_j) if $S_iA_i < S_jA_j$. □

Proposition 9. *The maximum load W^* that can be processed in time T can be found in $O(m \min\{\lfloor T \rfloor, \sum_{i=1}^{p} S_i\})$ time if $(C_i)_{1\leqslant i\leqslant p} = 0$.*

Proof. Let us assume that some sequence of worker activation is fixed, and without loss of generality that it is $P_1, \ldots, P_p$. We only have to choose a subset of the workers. The amount of load that can be processed by P_i in time T, provided that it finishes communications at time $\tau \geqslant S_i$ and that $C_i = 0$, is $W_i = \max\{0, \frac{T-\tau}{A_i}\}$. W^* can be calculated via function $W(i, \tau)$, which is the maximum load amount processed by workers selected from set $\{P_1, \ldots, P_i\}$ finishing communications at time τ, for $i = 1, \ldots, p$ and $\tau = 1, \ldots, \min\{\lfloor T \rfloor, \sum_{i=1}^{p} S_i\}$. $W(i, \tau)$ can be calculated in $O(p \min\{\lfloor T \rfloor, \sum_{i=1}^{p} S_i\})$ using the following recursive equations:

$$W(i,\tau) = \begin{cases} W(i-1,\tau) & \text{for } \tau < S_i \\ \max \begin{cases} W(i-1,\tau), \\ W(i-1,\tau - S_i) + \max\{0, \frac{T-\tau}{A_i}\} \end{cases} & \text{for } \tau \geqslant S_i \end{cases}$$

for $i = 1, \ldots, p, \tau = 1, \ldots, \min\{\lfloor T \rfloor, \sum_{i=1}^{p} S_i\}$. $W(j,0) = 0$ for $j = 0, \ldots, p$. $W(0,\tau) = 0$, for $\tau = 1, \ldots, \min\{\lfloor T \rfloor, \sum_{i=1}^{p} S_i\}$. The maximum load is $W^* = \max_{1 \leqslant \tau \leqslant \min\{\lfloor T \rfloor, \sum_{i=1}^{p} S_i\}} W(m,\tau)$. Let τ^* satisfy $W^* = W(p,\tau^*)$. The set of workers taking part in the computation can be found by backtracking from $W(p,\tau^*)$ and selecting those workers P_i for which $W(i,\tau) = W(i-1, \tau - S_i) + \max\{0, \frac{T-\tau}{A_i}\}$. □

Theorem 10. *The minimum schedule length T^* for a given load W can be found in $O((\log W + \log p + \log A_{max} + \log S_{max}) m \min\{\lfloor \max_i S_i + W A_{max} \rfloor, \sum_{i=1}^{p} S_i\})$ time if $(C_i)_{1 \leqslant i \leqslant p} = 0$.*

Proof. Let $A_{min} = \min_i\{A_i\}, A_{max} = \max_i\{A_i\}$. In the optimum sequence workers are activated according to the nondecreasing order of $S_i A_i$ by Proposition 8. For a given schedule length T, the maximum problem size W^* can be found in $O(p \min\{\lfloor T \rfloor, \sum_{i=1}^{p} S_i\})$ time according to Proposition 9. The minimum schedule length can be found by a binary search over the values of T. It remains to show that the number of the calls to the pseudopolynomial time algorithm is limited.

Let $x_i = 1$ if P_i takes part in the computation, and $x_i = 0$ otherwise. Thus, vector $\overline{x} = [x_1, \ldots, x_p]$ represents a subset of $\{P_1, \ldots, P_p\}$, the workers that take part in the computation. The load amount that can be processoed in time T is

$$W = \sum_{i=1}^{p} \frac{T x_i}{A_i} - \sum_{i=1}^{p} \sum_{j=i}^{p} \frac{x_i x_j S_i}{A_j}. \tag{8}$$

$\frac{T x_i}{A_i}$ is the amount of load that could have been processed provided that there were no communication delays. $x_i S_i$ is the computation time lost due to the activation of P_i. This loss of computing time affects workers P_j for $j \geqslant i$ because the activation sequence is fixed. Hence, $\sum_{i=1}^{p} \sum_{j=i}^{p} \frac{x_j x_i S_i}{A_j}$ is the amount of load lost due to the communication delays. It follows from equation (8) that W is a piecewise-linear nondecreasing function of T as required by Theorem 1. Therefore, the optimum T^* for a given W is a point on one segment or on an intersection of two segments of this piecewise-linear function. Let $\overline{x}$, and $\overline{x}'$ represent two different subsets of workers for which W is maximum at two different schedule lengths. The two linear functions of load amounts that can be processed in time T by workers corresponding to $\overline{x}$, and $\overline{x}'$ intersect at

$$T(\overline{x}, \overline{x}') = \frac{\sum_{i=1}^{p} \sum_{j=i}^{p} (x_i x_j - x'_i x'_j) \frac{S_i}{A_j}}{\sum_{i=1}^{p} \frac{x_i - x'_i}{A_i}} \tag{9}$$

Thus, the minimum distance between two different intersections is $\lambda = \frac{1}{A_{max} \sum_{i=1}^{p} \frac{1}{A_i}} \geqslant \frac{A_{min}}{p A_{max}} \geqslant \frac{1}{p A_{max}}$. If the difference between two values of T_1, T_2 visited in the binary search is smaller than λ, then either the same subset of workers gives maximum load for T_1 and T_2 or two different subsets of workers are selected for T_1, T_2. In the first case T^* can be found using linear interpolation of function (8). In the second case there is one more intersection T_3 between T_1, T_2, which can be found using (9), and then T^* can be found using linear interpolation either to the left or to the right of T_3. Since no schedule is longer than $S_{max} + W A_{max}$ and the resolution is λ, the binary search for T^* over T values can be terminated in $O(\log((S_{max} + W A_{max}) A_{max} p)) = O(\log W + \log p + \log A_{max} + \log S_{max})$ steps. The complexity of the whole algorithm is not greater than $O((\log W + \log p + \log A_{max} + \log S_{max}) p \min\{\lfloor S_{max} + W A_{max} \rfloor, \sum_{i=1}^{p} S_i\})$. □

4.4 Complexity considerations

It follows from Section 4.2 that DLS is NP-hard, because its special case DLS{1Round, C_i =0} is NP-complete (see Figure 4). However, proving that DLS belongs to NP is quite difficult. Indeed, as we need to know for each activation the amount of data that is sent to each host, the complexity of a *reasonable* DLS certificate, i.e., the length of the string encoding a solution, will be at least $\Omega(act_{max})$ (one needs at least to encode the activation sequence).

Theorem 11. *The optimal number of activations act_{max} for DLS-Opt$T(W)$ is $\Omega(\sqrt{W})$ for some instances.*

Proof. We consider the simple instance with $p = 1$, $(S_1, C_1, A_1) = (1, 1, 1)$. The only possible activation sequences are thus $(1), (1,1), (1,1,1), \ldots$. Let us denote by $T_n(W)$ the time needed to process W units of load when using n activations without idle time. We can easily prove that

$$T_n(W) = \frac{n+1}{2} + \frac{n+1}{n} W$$

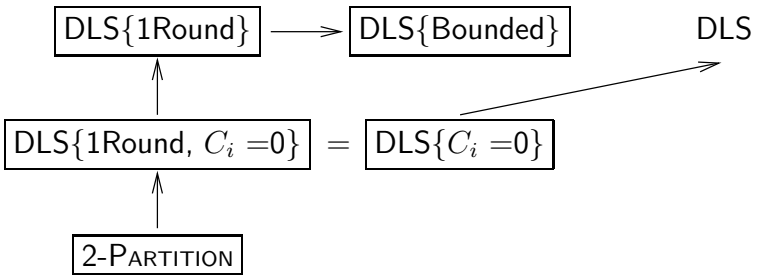


Figure 4: Complexity hierarchy. Boxed problems belong to NP and arrows depict the NP-harder relation.

Thus, we have $T^*(W) = min_n T_n(W)$ and we know that T^* is a continuous piecewise-linear function. Therefore, slope modifications of T^* occur for $T_n(W) = T_{n+1}(W)$ i.e. for $W = \frac{n^2+n}{2}$. Therefore the optimal number of activation is $\Theta(\sqrt{W})$. □

The previous theorem means that there is no polynomial bound on act_{max}. Intuitively, the optimal number of activations may grow roughly linearly with $\sqrt{W}$. Therefore, exponentially in the problem size because W is encoded with $O(\log W)$ bits. If act_{max} were bounded by polynomial fractions in W, the A_i's, C_i's and S_i's, then the complexity of our certificate would be $O(q(W, A_i, C_i, S_i))$. However, for problem DLS to be in NP the certificate length should be $O(r(\log(W), \log(A_i), \log(C_i), \log(S_i)))$, where q, r are polynomials. In other words, the number of activations needed to reach the optimum may be far to big. It does not really prove that DLS probably does not belong to NP because we could use another kind of certificate. But it also seems hard to have a certificate smaller than the activation sequence as the fraction of load sent at each activation seems mandatory.

As an alternative, we let the input include a bound on the number of activations. Such a hypothesis makes sense in practice as an arbitrarily complex schedule may not be desirable. We define problem DLS{Bounded} where we enforce that the maximum number of activations is bounded by the log of a bound K.

Problem 6 (DLS{Bounded}). *Given W, p workers, $(A_i)_{1\leqslant i\leqslant p}$, $(S_i)_{1\leqslant i\leqslant p}$, $(C_i)_{1\leqslant i\leqslant p}$, a rational number $T \geqslant 0$, and an integer K, is it possible to compute all W load units with at most $\log(K)$ activations within T time units after the master starts sending out the first load unit?*

DLS{Bounded} clearly belongs to NP. DLS{1Round, C_i =0} being a particular instance of DLS{Bounded}, DLS{Bounded} is NP-complete as well. It should however be noted that, unlike many other problems (e.g., finding broadcast trees optimizing network throughput [5]), NP-hardness does not come from the bound on the number of activations, but from the resource selection problem. This bound on the number of activations is mainly an artifact of defining our problem in NP.

5 Exact algrithms

5.1 Mixed Integer Linear Programming

Let us assume that we set a bound act_{max} on the number of activations that can be performed. Let $\chi_i^{(j)}$ be a binary value indicating whether worker i is used at activation j. Let us recall that $\alpha_i^{(j)}$ denotes the size of the j^{th} chunk of load sent to worker i. Then our problem can be formulated as a mixed

optimization program:

$$\begin{array}{l}\text{MINIMIZE} \quad T \quad , \\ \text{UNDER THE CONSTRAINTS} \\ \left\{\begin{array}{ll} \text{(10a)} & \displaystyle\sum_{j=1}^{act_{max}} \sum_{i=1}^{p} \chi_i^{(j)} \alpha_i^{(j)} = W \\ \text{(10b)} & \displaystyle \forall k \leqslant act_{max}, \forall l : \left(\sum_{j=1}^{k} \sum_{i=1}^{p} \chi_i^{(j)} (S_i + \alpha_i^{(j)} C_i) \right) + \sum_{j=k}^{act_{max}} \chi_l^{(j)} \alpha_l^{(j)} A_l \leqslant T \\ \text{(10c)} & \displaystyle \forall k \leqslant act_{max} : \sum_{i=1}^{p} \chi_i^{(k)} \leqslant 1 \\ \text{(10d)} & \forall i,j : \chi_i^{(j)} \in \{0,1\} \\ \text{(10e)} & \forall i,j : \alpha_i^{(j)} \geqslant 0 \end{array}\right. \end{array} \tag{10}$$

Constraint (10a) enforces the fact that the whole workload W is processed by our workers. Constraint (10c) impose that at most one worker is used in each activation. The leftmost part of Constraint (10b) represents the time at which the k^{th} communication ends and the middle one is a lower bound on the computation time of worker l after the k^{th} activation. The sum of these two times has thus to be smaller than the makespan T. Considering in backward order the activations where a given worker l is used, it is not hard to see from the constraints that one will obtain a feasible schedule [16].

As such, this program is not linear. However, it can easily be transformed into an equivalent Mixed Integer Linear Program (MILP) by introducing a new variable as follows:

$$\begin{array}{l}\text{MINIMIZE} \quad T \quad , \\ \text{UNDER THE CONSTRAINTS} \\ \left\{\begin{array}{ll} \text{(11a)} & \displaystyle\sum_{j=1}^{act_{max}} \sum_{i=1}^{p} \beta_i^{(j)} = W \\ \text{(11b)} & \displaystyle \forall k \leqslant act_{max}, \forall l : \left(\sum_{j=1}^{k} \sum_{i=1}^{p} \chi_i^{(j)} S_i + \beta_i^{(j)} C_i \right) + \sum_{j=k}^{act_{max}} \beta_l^{(j)} A_l \leqslant T \\ \text{(11c)} & \displaystyle \forall k \leqslant act_{max} : \sum_{i=1}^{p} \chi_i^{(k)} \leqslant 1 \\ \text{(11d)} & \forall i,j : \chi_i^{(j)} \in \{0,1\} \\ \text{(11e)} & \forall i,j : 0 \leqslant \beta_i^{(j)} \leqslant \chi_i^{(j)} W \end{array}\right. \end{array} \tag{11}$$

This MILP can be seen as a polynomial certificate to the DLS{Bounded} problem where at most act_{max} activations are allowed. Such a program can easily be solved by using a branch and bound technique since the linear relaxation of the $\chi_i^{(j)}$'s provides a lower bound on the solution of the original problem. In the rest of this article, we call BB the branch-and-bound algorithm that solves this MILP.

5.2 Lighter Branch and Bound

In this section we present a branch and bound algorithm, which we call BB-LIGHT. It is based on the optimality principle decreeing that there is idle time neither in communications, nor in computations, as proved in Section 3. A branch and bound algorithm consists of two components: a branching scheme that guides the exhaustive search of the solution space, and a bounding method that prunes the search space.

The branching scheme divides the search space into subsets which are either eliminated as certainly not providing an optimum solution, or divided into subsets to be further examined. The process of search space examination can be viewed as constructing of a tree. Each tree node represents a subset of solutions. In our case the search space consists of the sequences of communications to the workers. The algorithm starts with an empty communication sequence. The sequences are constructed by appending a new worker to some already existing sequence. For example, some sequence $(P_a, \ldots, P_z)$ of length l is expanded by appending all possible workers on the position $l+1$. Thus, the node representing sequence

$(P_a, \ldots, P_z)$ is branched into sequences $(P_a, \ldots, P_z, P_1), \ldots, (P_a, \ldots, P_z, P_p)$. Note that each sequence is a potential solution, for which distribution of the load and schedule length have to be calculated. These are obtained from a linear system (1a), (1b), under the assumptions that these constraints are satisfied with equality, and all α_js are positive ((1c) is satisfied). Under these assumptions loads α_j may be calculated as a solution of a system of linear equations (1a), (1b) rather than by using linear programming. An infeasible sequence is recognized if some $\alpha_j < 0$ (in other words (1c) is not satisfied). The tree is searched in the depth-first order. An upper limit act_{max} is imposed on the depth of the tree.

The bounding scheme eliminates solution subsets, i.e., tree nodes, for which there is no chance of obtaining a better solution than the best one already known. The quality of the node is evaluated by calculation of a lower bound on the length of the schedules constructed with its successors. Therefore a sequence $(P_a, \ldots, P_z)$ represents all the sequences starting with the sequence $(P_a, \ldots, P_z)$. The lower bound is calculated as follows. A minimum workload W_1 needed to keep processors in the sequence $(P_a, \ldots, P_z)$ working is calculated using *equations* (1a), (1b) with the additional assumption that $\alpha_z^{(l)} = 0$. From the same linear system, schedule length T_1 can be found for load W_1. The remaining load $W - W_1$ must be sent to the workers in time at least $T_2 = \min_{1 \leqslant i \leqslant p}\{C_i\}(W - W_1)$. In this time workers may compute at most $W_2 = T_2 \sum_{i=1}^{p} \frac{1}{A_i}$ load units. The remaining load must be processed in time at least $T_3 = \max\{0, \frac{W - W_1 - W_2}{\sum_{i=1}^{p} \frac{1}{A_i}}\}$. Hence, the lower bound is $T_1 + T_2 + T_3$.

Note that the lighter branch and bound proposed here still has exponential worst case running time, but it is not using linear programming.

6 Heuristics

In this section we present several scheduling heuristics to solve the general DLS problem. For each heuristic we quantify the trade-off between the time to compute the schedule and the quality of the schedule. We explore a range of heuristics, from simplistic and fast to sophisticated and potentially time consuming.

Most heuristics in this section attempt to determine a good activation sequence, and then compute the best chunk sizes. Therefore, these heuristics solve (10) for one or many activation sequences. Although (10) may be solved faster in some cases by taking advantage of peculirities of the activation sequence, we use a generic linear program solver. Indeed, solving a linear program is very fast (a few miliseconds on a standard CPU) and thus does not lead to prohibitively long schedule computation times. Once the chunk sizes and computed, and given an activation sequence, the makespan can be easily computed.

6.1 Simple Heuristics

Also seen in Section 4.1, DLS{1-round,S_i =0} can be solved by sorting processors by increasing C_i's in the activation sequence. Likewise, if one assumes that the total load is "sufficiently large", workers should be sorted by increasing C_i's just like when all S_i's are zero. Trying to cyclicly use all processors sorted by communication time is thus a natural heuristic. More formally, if we assume the $C_1 \leqslant C_2 \leqslant \cdots \leqslant C_p$, we compute the optimal makespan T_k associated to the activation sequences γ_k, which consists k repetitions of the p sorted processors: $\gamma_k = \{\underbrace{1, \ldots, p, 1, \ldots, p, \ldots, 1, \ldots, p}_{k \text{ times } \{1, \ldots, p\}}\}$.

Note that if a processor receives 0 units of load in a round (as computed when solving (10)), we remove it from the activation sequence and its latency is therefore not counted in the makespan. We stop as soon as adding a new round does not lead to any improvement (i.e., as soon as $T_{k+1} \geqslant T_k$). We call this heuristic Communication-First. For the sake of completeness, we also developped Computation-First and Latency-First in which processors are sorted by increasing A_i's and increasing S_i's, respectively.

6.2 Genetic Algorithm

Genetic algorithms are randomized search methods that apply genetic operators on a pool of solutions with the goal of discovering the optimum. Genetic algorithms are widely used in solving discrete optimization problems, and details on various implementations and applications can be found, e.g., in [23, 26]. Here we only outline details of our implementation. Results of preliminary application of genetic algorithms on the DLS problem can be found in [17].

A solution is an activation sequence, which is encoded as a string of workers. The string has a predetermined length act_{max}. The quality of the solution is determined as schedule length T by the

linear program derived from (10) for a fixed communication sequence. A pool of G random solutions is generated on which the genetic operators of crossover, mutation, and selection are iteratively applied. Single point crossover has been applied to generate Gp_C new solutions. Mutation changes $Gact_{max}p_M$ randomly selected destinations in the whole population. Selection chooses for the new population the best half of the old population, and for the second half of the population newly generated solutions are chosen using roulette wheel strategy. The above genetic operators are iteratively applied to construct new populations until an upper limit U_T on the number of iterations is reached. The algorithm also stops after reaching an upper limit U_O of the iterations without improving the quality of the best solution.

Parameters G, p_C, p_M, U_T, U_O were selected in the following way [17]. A set of 100 random instances were generated and solved by genetic algorithm. The measure of quality of tuning was the sum of the schedule lengths for all the instances, and the rate of its convergence with the iteration number. Note that this criterion is equivalent to the average relative distance from the optimum, but the actual optima need not be known. $G = 50$ was selected first, then $p_C = 0.8, p_M = 0.03$, and for these fixed parameters $U_T = 100, U_O = 10$ were finally chosen. We call the resulting algorithm GA.

6.3 Uniform Multi-Round (UMR)

In this section we briefly describe the UMR (Uniform Multi-Round) algorithm presented in [32], which has been specifically designed to schedule divisible loads. The idea behind UMR is simple: assign chunks of "uniform" sizes to all workers within each round, increasing the chunk size between rounds geometrically. Here "uniform" means that it takes the same amount of time for each worker to compute its chunk at each round (i.e., the product $\alpha_i^{(j)} A_i = X_j$ does not depend on i).

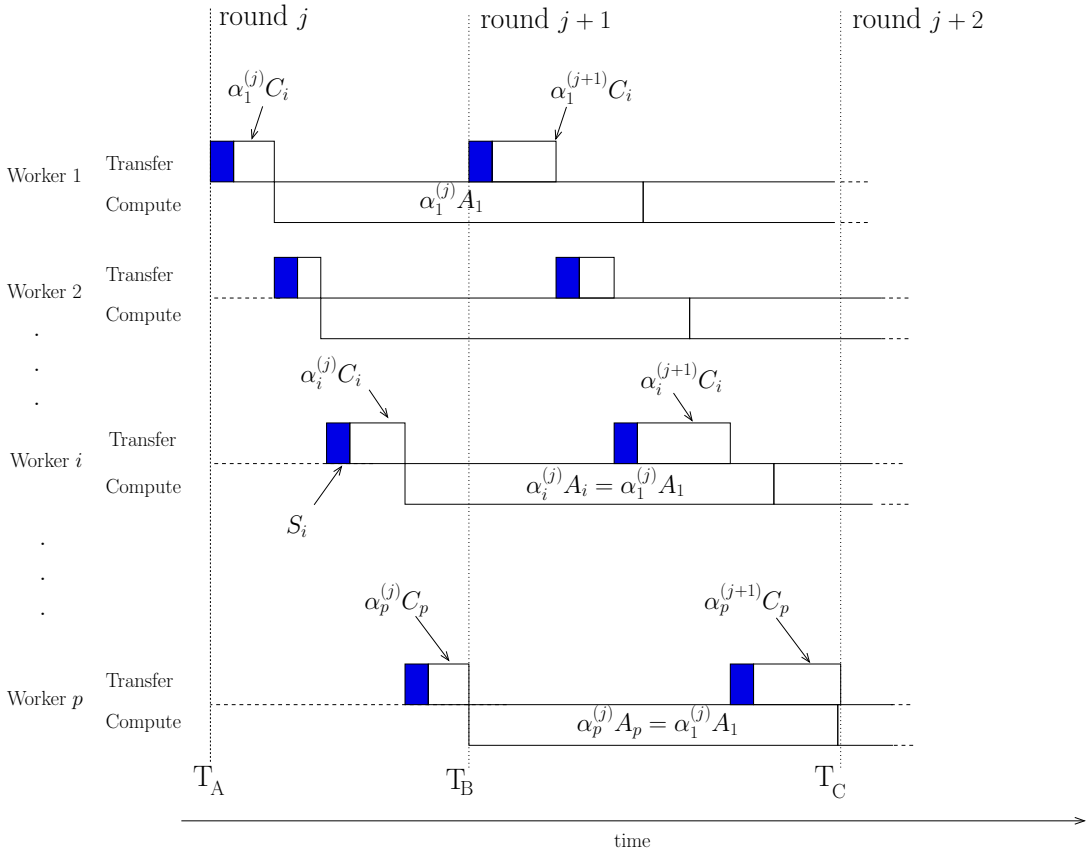


Figure 5: Multi-round schedule with UMR.

For illustration purposes, a UMR schedule is depicted in Figure 5 for a heterogeneous platform. Time is shown on the x-axis, and workers are shown on the y-axis. The computation start-up costs S_i are shown in dark grey boxes. We can see that the computation times of chunks are identical across all workers within each round (chunk dispatching for round j in the figure goes from time T_A to time T_B).

In order to maximize network utilization, UMR imposes that the *last* worker receives data for round $j+1$ exactly when it finishes its computation for round j, as seen on the figure. Such a condition is

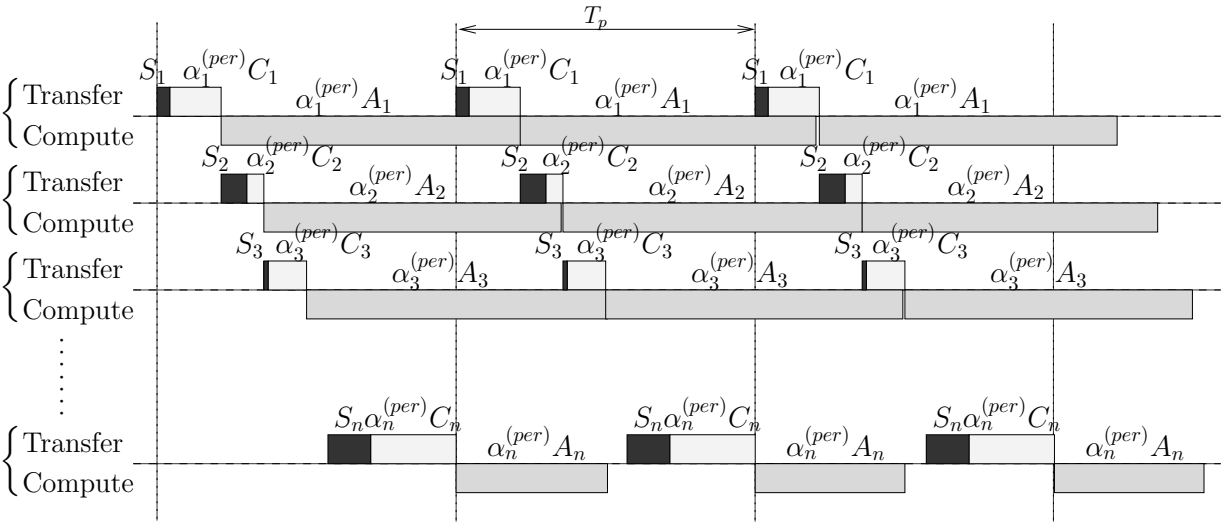


Figure 6: Sketch of a periodic multi-round schedule using the first n workers P_1 to P_n, where $n \leqslant p$.

imposed for all workers in the Multi-Installment algorithm [8], which is unfortunately only applicable to homogeneous platforms. In this sense, UMR can be seen as a relaxation of Multi-Installment so that it can be applied to heterogeneous platforms. The above condition can be written as:

$$\alpha_i^{(j)} A_i = \sum_{k=1}^{p} (S_i k + \alpha_k^{(j+1)} C_i k).$$

With the above equation, along with the facts that $\alpha_i^{(j)} A_i$ does not depend on i and that the sum of all chunk sizes sums up to the entire load, makes it possible to compute the chunk sizes recursively (they increase geometrically at each round). We refer the reader to [31] for all details. Note that in the last round UMR decreases chunk sizes within the round so that all workers finish computing at the same time.

As seen earlier, a difficult issue is that of resource selection. UMR uses a simple heuristic that is inspired by the work in [1]: workers with faster networks (i.e., higher bandwidths) are selected first until no more worker can be used effectively.

6.4 Periodic

In this section, we briefly present the periodic asymptotically optimal algorithm from Beaumont *et al.* [2]. An algorithm is asymptotically optimal if the ratio of the time to execute a workload W over the optimal time to execute this workload tends to 1 as W tends to infinity. The sketch of the algorithm is as follows. The overall processing time T is divided into k regular periods of duration T_p (see Figure 6). Let us consider the following linear program:

$$\begin{array}{l} \textsc{Maximize} \quad \rho = \sum_{i=1}^{p} \beta_i, \\ \textsc{subject to} \\ \left\{ \begin{array}{l} \forall 1 \leqslant i \leqslant p, \quad 0 \leqslant \beta_i A_i \leqslant 1 \\ \sum_{i=1}^{p} \beta_i C_i \leqslant 1 \end{array} \right. \end{array}$$

This linear program provides an upper bound ρ on the throughput that any schedule can reach (latency, start-up and close-up of the schedule are ignored). Resource selection is automatically performed during the resolution of this linear program. The periodic schedule is built such that W/k units of load are processed each period. Using, the previously computed values for the β_i's and ρ, we define $\alpha_i^{(per)} = \frac{\beta_i}{\rho}\frac{W}{k}$. Hence we have:

$$T_p = \max\left(\sum_i \frac{\beta_i}{\rho}\frac{W}{k}C_i + S_i, \max_i\left(\frac{\beta_i}{\rho}\frac{W}{k}A_i\right)\right) = \max\left(\frac{a}{k} + b, \frac{a'}{k}\right), \text{ where } \begin{cases} a = W.\sum_i \frac{\beta_i C_i}{\rho}, \\ b = \sum_i S_i, \text{ and} \\ a' = W.\max_i \frac{\beta_i A_i}{\rho} \end{cases}$$

Therefore:

- If $k > (a' - a)/b$ then $a/k + b > a'/k$ and the makespan is:

$$T = (k+1)T_p = a + b + a/k + bk, \text{ which is minimized for } k = \sqrt{b/a}.$$

- If $k \leqslant (a' - a)/b$ then $a/k + b < a'/k$ and the makespan is:

$$T = (k+1)T_p = a' + a'/k, \text{ which is minimized for } k \text{ as large as possible.}$$

The "optimal" number of round for such a periodic schedule is therefore $k = \max(\sqrt{b/a}, (a' - a)/b)$. We call this algorithm Periodic.

One drawback of the schedule computed by Periodic is that it is very rigid: the exact same amount of workload is sent to each worker during every period. Intuitively, rounds should be smaller in the beginning to allow a better overlap of communications and computations and a better start-up time (like with the heuristics described in Section 6.3). This is why we also propose a variant, Periodic-Optimized, that uses the exact same activation sequence, but computes the optimal values of the α_i's for this activation sequence by solving (10). The resulting schedule is likely to be more complex but also more efficient.

7 Conclusion

In this article, we have defined the multi-round divisible load scheduling problem and studied its complexity. We have stated and proved an optimality principle: *For an optimal activation sequence and the corresponding optimal load distribution in DLS-OptW and DLS-OptT, all messages, except maybe the trailing ones, convey some load and there is no idle time.* We have proved that this problem is NP-complete even for simple instances (infinite bandwidth) and we have proposed a pseudo-polynomial algorithm for these instances. We have discussed the belonging to NP and showed that the optimal number of activations is not polynomial in the inputs of DLS. We also have presented a wide survey of previously known or original techniques to solve this problem.

The complexity of a few problems remains open:

- Computing a DL schedule entails three steps: (i) select which workers should participate in the computation; (ii) decide in which order workers should receive load chunks and how many times; and (iii) compute how much work each load chunk should comprise. In essence, the NP-completeness results is based on the selection problem. We have seen that computing the chunks' sizes is easy. It would thus be interesting to know whether the problem is hard once the selection is done i.e., if the ordering problem is hard.

- We have seen that the belonging of the general DLS problem to NP is unclear. Particularly because the optimal number of activations is not polynomial in the inputs of DLS. In practice, it is of course important to have a description of the schedule. And unless some particular structure of the optimal activation sequence is shown, it is very unlikely that it belongs to NP. We do not know however yet how to prove that a problem does not belong to NP.

- DLS{1Round} clearly belongs to NP but we have only be able to prove its weak NP-completeness. The question whether it is strongly NP-complete or not remains open.

References

[1] O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert. Bandwidth-Centric Allocation of Independent Tasks on Heterogeneous Platforms. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, pages 67–72, June 2002.

[2] O. Beaumont, H. Casanova, A. Legrand, Y. Robert, and Y. Yang. Scheduling Divisible Loads on Star and Tree Networks: Results and Open Problems. *IEEE Transactions on Parallel and Distributed Systems*, 16(3):207–218, 2005.

[3] O. Beaumont, A. Legrand, L. Marchal, and Y. Robert. Independent and Divisible Task Scheduling on Heterogeneous Star-Shaped Platforms with Limited Memory. Technical Report RR2004-22, Ecole Normale Superieure de Lyon, April 2004.

[4] O. Beaumont, A. Legrand, and Y. Robert. Optimal Algorithms for Scheduling Divisible Workloads on Heterogeneous Systems. Technical Report 2002-36, Ecole Normale Superieure de Lyon, October 2002.

[5] O. Beaumont, L. Marchal, and Y. Robert. Complexity results for collective communications on heterogeneous platforms. *Int. Journal of High Performance Computing Applications*, 20(1):5–17, 2006.

[6] W. Bethel, B. Tierney, J. lee, D. Gunter, and S. Lau. Using high-speed WANs and network data caches to enable remote and distributed visualization. In *Proceedings of Supercomputing (SC'00)*, 2000.

[7] V. Bharadwaj, D. Ghose, and V. Mani. Optimal Sequencing and Arrangement in Single-Level Tree Networks With Communication Delays. *IEEE Transactions on Parallel and Distributed Systems*, 5(9):968–976, 1994.

[8] V. Bharadwaj, D. Ghose, and V. Mani. Multi-Installment Load Distribution in Tree Networks With Delays,. *IEEE Trans. on Aerospace and Electronc Systems*, 31(2):555–567, 1995.

[9] V. Bharadwaj, D. Ghose, V. Mani, and T. G. Robertazzi. *Scheduling Divisible Loads in Parallel and Distributed Systems*. IEEE Computer Society Press, 1996.

[10] V. Bharadwaj, D. Ghose, and T. Robertazzi. A new paradigm for load scheduling in distributed systems. *Cluster Computing*, 6(1):7–18, 2003.

[11] V. Bharadwaj and S. Ranganath. Theoretical and Experimental Study on Large Size Image Processing Applications Using Divisible Load Paradigm on Distributed Bus Networks. *Image and Vision Computing*, 20(13-14):917–1034, 2002.

[12] J. Blazewicz and M. Drozdowski. Distributed Processing of Divisible Jobs With Communication Startup Costs. *Discrete Applied Mathematics*, 76:21–41, 1997.

[13] T. Braun, H. Siegel, and N. Beck. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing*, 61:810–837, 2001.

[14] Y.-J. Chiang, R. Farias, C. T. Silva, and B. Wei. A unified infrastructure for parallel out-of-core isosurface extraction and volume rendering of unstructured grids. In *Proceedings of the IEEE Symposium on Parallel and Large-data Visualization and Graphics*, pages 59–66, 2001.

[15] E. Coffman Jr., M. Garey, and D. Johnson. Bin Packing Approximation Algorithms: A Survey. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS Publishing Co., Boston, MA., 1996.

[16] M. Drozdowski. *Selected problems of scheduling tasks in multiprocessor computing systems*. Poznań University of Technology, Poznań, Poland, 1997.

[17] M. Drozdowski and M. Lawenda. On optimum multi-installment divisible load processing in heterogeneous distributed systems. In J.C.Cunha and P.D.Medeiros, editors, *Proceedings of Euro-Par 2005, LNCS*, volume 3648, pages 231–240. Springer-Verlag, 2005.

[18] M. Drozdowski and P. Wolniewicz. Experiments With Scheduling Divisible Tasks in Clusters of Workstations. In A.Bode, T.Ludwig, W.Karl, and R.Wismüler, editors, *Proceedings of Europar 2000, LNCS*, volume 1900, pages 311–319. Springer-Verlag, 2000.

[19] M. Drozdowski and P. Wolniewicz. On the Complexity of Divisible Job Scheduling with Limited Memory Buffers. Technical Report RA-001-2001, Institute of Computing Science, Poznań University of Technology, 2001.

[20] I. Foster and C. Kesselman, editors. *Grid 2: Blueprint for a New Computing Infrastructure*. M. Kaufmann Publishers, 2nd edition edition, 2003.

[21] A. Garcia and H.-W. Shen. Parallel volume rendering: An interleaved parallel volume renderer with PC-clusters. In *Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 51–59, 2002.

[22] M. R. Garey and D. S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W. H. Freeman, 1991.

[23] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.

[24] Y. Kwok and I. Ahmad. Benchmarking and Comparison of the Task Graph Scheduling Algorithms. *Journal of Parallel and Distributed Computing*, 59(3):381–422, 1999.

[25] C. Lee and M. Hamdia. Parallel Image Processing Applications on a Network of Workstation. *Parallel Computing*, 21:137–160, 1995.

[26] Z. Michalewicz. *Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 1996.

[27] G. Miller, D. G. Payne, T. N. Phung, H. Siegel, and R. Williams. Parallel Processing of Spaceborne Imaging Radar Data. In *Proceedings from Supercomputing (SC'95)*, 1995.

[28] P.Wolniewicz. *Divisible Job Scheduling in Systems with Limited Memory*. PhD thesis, Poznań University of Technology, Instiute of Computing Science, 2003.

[29] T. Robertazzi. Ten reasons to use divisible load theory. *IEEE Computer*, 36(5):63–68, 2003.

[30] Visible human project. http://www.nlm.nih.gov/research/visible/visible_human.html.

[31] Y. Yang. *Scheduling Divisible Loads in Multiple Rounds*. PhD thesis, University of California at San Diego, Department of Computer Science and Engineering, July 2005.

[32] Y. Yang, K. V. de Raadt, and H. Casanova. Multi-round algorithms for scheduling divisible loads. *IEEE Transactions on Parallel and Distributed Systems*, 16(11):1092–1102, 2005.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399